**SMS SPAM CLASSIFIER**

**Submitted for**

**Statistical Machine Learning CSET211**

Submitted by:

**(E23CSEU0321) ANOUSHA SINGH**

Submitted to

**DR. AMIT SONI**

**July-Dec 2024**

**SCHOOL OF COMPUTER SCIENCE AND ENGINEERING**

A close-up of a logo

Description automatically generated

**INDEX**

|  |  |  |
| --- | --- | --- |
| **Sr.No** | **Content** | **Page No** |
| 1. | Abstract | 3 |
| 2. | Introduction | 4 |
| 3. | Methodology | 5 |
| 4. | Hardware/Software Requirements | 7 |
| 5. | Experimental Result | 7 |
| 6. | Conclusion | 7 |
| 7. | Future scope | 8 |
| 8. | Github Link | 8 |

# Abstract

The "SMS Spam Classifier" project tackles the growing problem of spam messages in SMS communication, which is especially important because SMS is commonly used for critical updates like delivery notifications and One-Time Passwords (OTPs). When spam filters mislabel these important messages, it can cause major inconveniences for users and disrupt essential services. This project aims to develop a reliable spam classifier that can tell spam apart from genuine (ham) messages, ensuring users receive the messages they need without getting overwhelmed by unwanted content.

Using a machine learning approach, the classifier learns to recognize patterns in spam and ham messages, making it highly accurate in real-world conditions. With an easy-to-use interface, the classifier is practical for everyday use, allowing users and systems to detect spam in real-time without hassle. This project is designed to help in messaging apps, telecom services, and any system that needs reliable spam detection for better user experiences.

# Introduction

With SMS still widely used for communication, notifications, and verification codes, filtering out spam has become more important than ever. Every day, people receive countless messages, and a significant portion of them are spam—unsolicited or even malicious messages that clutter inboxes, waste time, and can sometimes pose security risks. These spam messages can be disruptive, and in cases where SMS is used for critical updates, like delivery notifications and One-Time Passwords (OTPs), missing or mislabeling these essential messages as spam can lead to real problems.

The "SMS Spam Classifier" project addresses this challenge by developing a machine learning model that can effectively distinguish between spam and genuine (ham) messages. Using a carefully selected dataset of SMS messages, the project applies natural language processing (NLP) techniques to analyze and understand the patterns that typically characterize spam and ham messages.

To improve the reliability and accuracy of the classifier, this project doesn’t rely on a single algorithm; instead, it combines multiple models using ensemble methods. This approach makes the classifier not only more accurate but also more adaptable to different kinds of SMS spam that might evolve over time.

Ultimately, the goal is to create a tool that is both effective and user-friendly, making it simple for anyone to filter SMS messages in real-time. This classifier has significant potential to improve the SMS experience, ensuring that important messages get through while keeping unwanted spam out.

# Methodology

From gathering and cleaning data to building an interface for end users, each stage is crucial for creating a reliable spam detection system.

1. **Data Collection and Preprocessing**:  
   The first step involves gathering of SMS dataset with a mix of spam and non-spam messages. Once we have this data, we need to make it ready for model training. This involves several preprocessing steps:

* **Text Cleaning**: The dataset is cleaned by removing any punctuation and special characters that don’t add much meaning. Duplicate entries or missing values are also removed.
* **Lowercasing**: All words are converted to lowercase to ensure that the same word in different cases is treated the same way.
* **Tokenization**: The text is split into individual words (tokens), which helps in analysing the frequency of specific words in spam and ham messages.
* **Stop Word Removal**: Common words like "and," "is," or "the" don’t usually help in classification, so we remove these to focus on words that carry more significance.
* **Stemming/Lemmatization**: Words are reduced to their base form, so "running," "runner," and "run" all map to the root "run." This process helps generalize the model, reducing noise in the data.

1. **Feature Extraction**:  
   After preprocessing, we need to convert the text data into a format that the machine learning model can work with. For this, we used Term Frequency-Inverse Document Frequency (TF-IDF) vectorization, which transforms the text into numerical vectors. TF-IDF helps by giving more importance to words that are frequent in spam or ham messages but less common overall, helping the model focus on words that distinguish spam from ham.
2. **Model Selection and Ensemble Technique**:  
   To build an accurate spam classifier, we experiment with several popular models. Each model has strengths in text classification, so we train and test multiple types, including:

* **Naive Bayes**: Known for its simplicity and effectiveness in text classification, it performs well when the data features are independent.
* **Random Forest**: An ensemble method that builds multiple decision trees and combines their outputs to improve accuracy and reduce overfitting.
* **K-Nearest Neighbors (KNN)**: A simple, instance-based learning algorithm that classifies messages by finding the most similar examples (neighbors) in the training data. It performs well when the data is well-structured and patterns are clear.

1. **User Interface Development**:  
   To make the SMS Spam Classifier easy to use, we developed a web interface using Python and the Streamlit library. The interface is user-friendly and interactive, allowing users to type or paste a message and receive an instant classification of whether the message is spam or ham with a single click. Python handles the backend logic, where the trained model processes the input message and generates the classification result. Streamlit provides an intuitive frontend, making the tool accessible and easy to deploy for personal use or integration into larger systems.

# Hardware/Software Required

To run this project, a system with at least 8GB of RAM and a 64-bit processor is required to run the model.

For software, the latest version of Python should be installed along with pip to manage dependencies. Essential libraries like NumPy, Pandas, Matplotlib, NLTK, Scikit-Learn, WordCloud, and Streamlit must also be installed to run the application smoothly.

An IDE such as Jupyter Notebook or VS Code is recommended for writing and executing the code, while a modern browser like Chrome or Edge is needed to run the application and access its interface. This setup ensures that the project runs efficiently and provides an optimal user experience.

# Experimental Results

The classifier was trained on a dataset of spam and ham messages, achieving an accuracy of near about 95%. The models demonstrated high precision and cross validation score, effectively identifying spam messages while minimizing false negatives. It performed well in classifying legitimate messages, such as OTPs and delivery notifications, ensuring reliable spam detection. The Streamlit-based user interface provided quick and accurate results, validating the system's efficiency and ease of use.

# Conclusion

The SMS Spam Classifier successfully addresses the problem of identifying spam messages with high accuracy and reliability. Among the models used, Naive Bayes and Random Forest emerged as the best performers for this classification task, achieving a precision of 1 and consistently high accuracy. These models demonstrated their effectiveness in distinguishing spam from legitimate messages, ensuring critical messages like OTPs and delivery updates are accurately classified. With a user-friendly interface built using Streamlit, the classifier is accessible and practical for real-world applications. This project underscores the capability of machine learning in enhancing communication security and streamlining SMS management.

# Future Scope

In the future, advanced methods like deep learning could be used to make the system even more accurate. A better classifier could be used or it could be upgraded to work with messages in different languages, making it useful for a wider audience.

Adding real-time spam detection for mobile apps or SMS services would make it more practical for everyday use. It could also be improved to identify new types of spam, like messages with hidden links or tricks to bypass detection. We can also add that whatever new messages the model is detecting as spam, we can use that as a dataset too so that the model becomes more refined.

# GitHub Link

<https://github.com/Anousha-Singh/SMS-Spam-Classifier>